Hindawi Publishing Corporation
BioMed Research International
Article ID 501528

Research Article

Building Integrated Ontological Knowledge Structures with Efficient Approximation Algorithms

Yang Xiang[1] and Sarath Chandra Janga[2]

[1]*Department of Biomedical Informatics, The Ohio State University, Columbus, OH 43210, USA*
[2]*Department of BioHealth Informatics, Indiana University-Purdue University Indianapolis, Indianapolis, IN 46202, USA*

Correspondence should be addressed to Sarath Chandra Janga; scjanga@iupui.edu

Received 22 October 2014; Revised 30 December 2014; Accepted 1 January 2015

Academic Editor: Dongchun Liang

The integration of ontologies builds knowledge structures which brings new understanding on existing terminologies and their associations. With the steady increase in the number of ontologies, automatic integration of ontologies is preferable over manual solutions in many applications. However, available works on ontology integration are largely heuristic without guarantees on the quality of the integration results. In this work, we focus on the integration of ontologies with hierarchical structures. We identified optimal structures in this problem and proposed optimal and efficient approximation algorithms for integrating a pair of ontologies. Furthermore, we extend the basic problem to address the integration of a large number of ontologies, and correspondingly we proposed an efficient approximation algorithm for integrating multiple ontologies. The empirical study on both real ontologies and synthetic data demonstrates the effectiveness of our proposed approaches. In addition, the results of integration between gene ontology and National Drug File Reference Terminology suggest that our method provides a novel way to perform association studies between biomedical terms.

1. Introduction

In recent years, ontologies are becoming increasingly important in knowledge engineering. Generally speaking, an ontology is a collection of concepts and their relations. It has wide applications in computer science and life science. For example, in computer science, semantic web uses web ontology language (OWL) to represent knowledge bases [1]. In life sciences, numerous important data structures and tools are built on ontologies. One of the most popular ontologies is gene ontology. Researchers use it frequently to measure the enrichment of gene clusters and to identify potential biomarkers. Two most famous ontology databases in the biomedical field are Unified Medical Language System (UMLS) [2] and NCBO BioPortal (https://bioportal.bioontology.org/). The former has more than 100 ontology datasets and the latter has more than 300 ontology datasets.

Although ontologies can be modeled as a directed graph, many ontologies are in fact hierarchical trees or have hierarchical tree-like structures. In the BioPortal website, users can find basic hierarchical properties of an ontology, such as the maximum depth and the maximum number of children. In the UMLS, the hierarchical structure of an ontology is documented in the "MRHIER.RRF" file with each line being a path from a term to its root. We can build a hierarchical tree from these paths by merging the common nodes starting from the root. Because the hierarchical structures of some ontologies are in fact directed acyclic graphs, the hierarchical tree may contain some duplicated concepts. To simplify our study, we treat them as independent concepts in this work.

An important knowledge discovery task is to identify knowledge associations. In life science, this task includes finding the associations between diseases and genes [3, 4] and between phenotypes and genotypes [5]. With the presence of ontologies, such a task has been extended from identifying the associations between terms to the associations between ontologies as a whole. For the latter, we should not only consider the term associations, but also the term associations in the context of their ontological structures. For example, if the parent and children of term a (a from ontology A) are

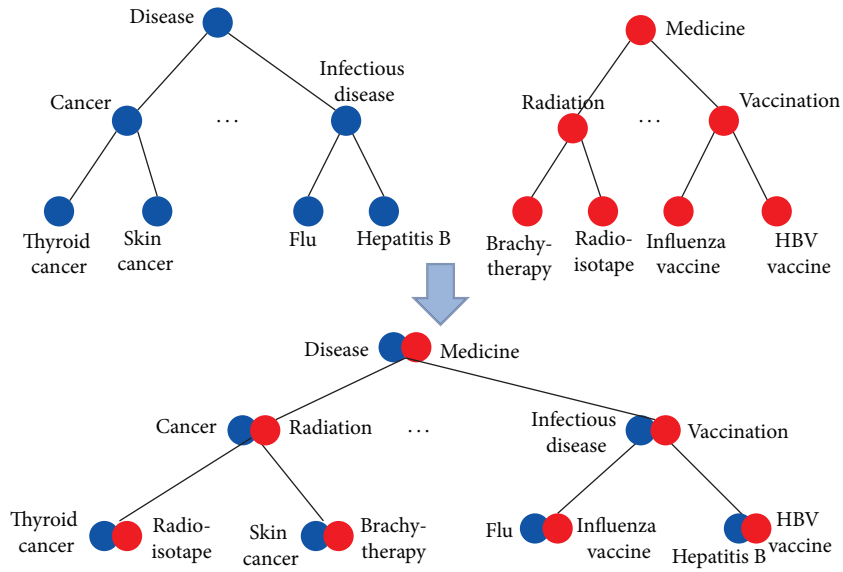


FIGURE 1: A simple example of integrating two hypothetical ontologies.

similar to those of term b (b from ontology B), Then a may be a good choice to be associated with b.

Early studies on ontology integration relied on domain experts to manually set up the integration rules [6]. However, this approach cannot meet the ever increasing volume of ontology datasets. Automatic ontology integration methods have been developed to address this issue. However, as we will see in the discussions of Section 1.3, these methods are often heuristic or have not demonstrated the effectiveness in integrating a large volume of ontology datasets. Thus, our goal is to develop an ontology integration method that is able to deliver optimal or close-to-optimal solutions for integrating a large volume of ontology datasets (particularly from the biomedical domain). As discussed above, we focus on ontologies with hierarchical tree-like structures which are often available in the biomedical domain. In addition, we assume the ontology term closeness measurement is available. This assumption is reasonable because many applications are able to identify ontology term similarities via additional data sources. For example, a closeness matrix between two sets of biomedical terms can be generated by using UMLS knowledge discovery methods such as KDLS [7]. Our problem can be formally described as follows.

1.1. Problem Formulation. The basic ontology integration problem in our work can be formulated as follows. Given ontology tree structures T_A and T_B and a closeness matrix M_{AB}, how can we efficiently generate an integrated ontology tree structure T_{AB} meeting the following two basic criteria?

(1) For any two vertices x and y in T_A (or T_B), the lowest common ancestor $\mathrm{LCA}_{T_A}(x, y)$ (or $\mathrm{LCA}_{T_B}(x, y)$) is contained by $\mathrm{LCA}_{T_{AB}}(x, y)$.

(2) It holds that $\mathrm{argmax}_{T_{AB}} f(T_{AB}) = \sum_{X \in V(T_{AB})} M_{AB}(X)$. Here $M_{AB}(X)$ is the entry value in the closeness matrix for the corresponding two vertices (one from T_A and the other from T_B) contained in the node X. $M_{AB}(X) = 0$ if X, a node of T_{AB}, contains only one vertex from T_A or T_B.

We name $f_{T_{AB}}$ the cohesion function of the integrated ontology T_{AB} and its value is the overall cohesion score of integrating T_A and T_B into T_{AB}. Correspondingly, we define function $g(T_A, T_B) = \max_{T_{AB}}(\sum_{v \in V(T_{AB})} M_{AB}(v))$ as the maximum cohesion function for integrating the ontologies T_A and T_B and its value is the maximum overall cohesion score (or, simply, maximum cohesion score). In a hierarchical ontology, the common part of any two terms can be described by their lowest common ancestor. For example, in Figure 1, flu and hepatitis B are both infectious diseases, and flu and cancer are both diseases. Thus, we use Criterion (1) to ensure that the basic logic of an ontology is preserved after integration.

An example of integrating two hypothetical ontologies that satisfy Criterion (1) is given in Figure 1. To facilitate the understanding of our problem definition, we also provide another example of integrating two ontologies in Figure 2. As we can see in Figure 2, the lowest common ancestor of nodes containing thyroid cancer and infectious disease is the node containing cancer instead of disease. We conclude that the integration is a violation of Criterion (1). In fact, we can easily see that there are multiple pairs of nodes with incorrect lowest common ancestors in Figure 2.

In Section 2.2, we will extend the basic problem definition to handle the integration of multiple (>2) ontologies. The two basic criteria will be extended correspondingly.

1.2. Main Contributions. We made the following major contributions in this work.

(i) We proposed a novel ontology integration problem that optimizes the cohesion function. We identified optimal structures in this problem and developed

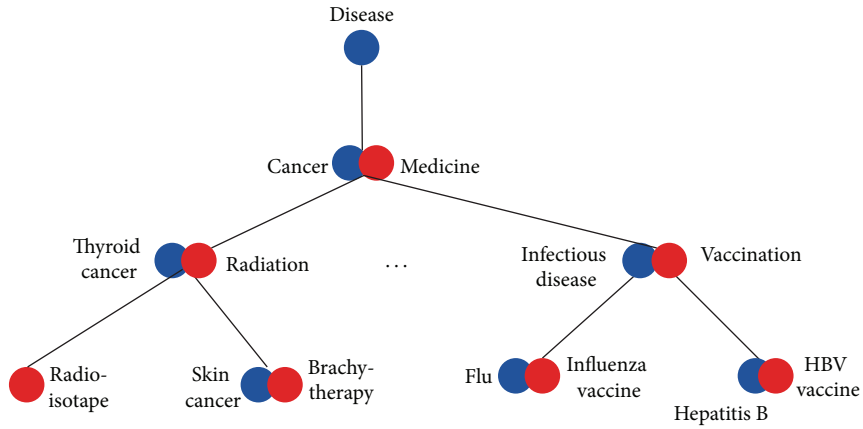


Figure 2: Another example of integrating the two ontologies in Figure 1. This integration violates Criterion (1).

optimal as well as efficient approximation solutions for this problem.

(ii) We extended the basic problem to handle the integration of large number of ontologies, and we developed both greedy and fast approximation algorithms for the extended problem.

(iii) We studied the proposed algorithms on both real and synthetic datasets and confirmed their effectiveness in integrating large volume of ontology datasets.

1.3. Related Work. Automatic ontology generation and integration are desirable in many applications and have been studied in the past decade. Although available methods for automatic ontology generation produce ontologies from a given type of data, such as gene networks [8], textual data [9], dictionary [10], and schemata [11, 12], they do not contribute to the integration of different types of ontologies which will bring innovative results on annotation/knowledge reuse and association studies. To address this issue, a number of studies have been focused on ontology integration [13, 14] and its medical domain applications [15]. The ontology integration methods available and used in these works can be generally classified into three categories.

Manually or Semiautomatic Setups. In [6], the authors presented a methodology for ontology integration by custom-tailored integration operations which include algebraic-specified 39 basic operations and 12 nonbasic operations derived from them. The authors identified a set of criteria, such as modularize, specialize, and diversify each hierarchy, for guiding the knowledge integration. In [16], the authors designed a semiautomatic approach for ontology merging. The ontology developers will be assisted by the system and guided to tasks needing their interventions.

Using Machine Learning Methods. Reference [17] describes an ontology mapping system GLUE that uses machine learning techniques for building ontology mappings. Specifically, GLUE uses multiple learning strategies. Each type of information from the ontologies is handled by a different learning strategy. The authors demonstrate that GLUE works effectively on taxonomy ontologies. Similarly, [18] also used multistrategy learning in matching pair identification. However, the ontologies used in the experiments of [18] contain less than 10 nodes. Although [17] studied the integration of larger ontologies in the experiments, those ontologies contain only around 34 to 176 nodes, much smaller than many ontologies used in the biomedical field.

Using Heuristic Approaches. Many automatic ontology integration methods [19, 20] fall into this category. They perform ontology integrations by using heuristic approaches from different perspectives. For example, [20] uses heuristic policies for selecting axioms and candidate merging pairs. From a quite different angle, [19] uses view-based query for guiding the ontology integrations.

These methods have a few major weaknesses including (1) lack of a systematic measurement to quantify the goodness for the ontology integration; (2) being generally heuristic with no theoretical results to show that the proposed integration approach is globally optimal or close to optimal; (3) being not for integrating large volume of ontologies. These weaknesses motivated us to develop efficient and near optimal solutions for integrating large ontology datasets.

2. Methods

2.1. Integrating a Pair of Ontologies. In this section, we focus on the basic problem of integrating two ontologies as formulated in Section 1.1. We will prove optimal structures in the problem and propose an optimal and an efficient approximation solution for this problem. These solutions are also the basis for solving the problem of integrating a large number of ontologies as described in Section 2.2.

2.1.1. Brutal-Force and Heuristic Solutions. Given Criteria (1) and (2), a brutal-force approach will pick up a best solution from all the solutions that start with integration involving at least one of the roots of the two ontology trees and iteratively integrate their descendants. Considering an extreme case where each ontology tree is a path of n vertices, we

```
push (0, 0) into queue q; {Integrating virtual roots of the two ontologies}
while |q| > 0 do
  (a, b) = pop(q);
  for i ∈ children of a on T_A do
    max_score = −1;
    to_merge = NULL;
    to_merge_root = NULL;
    for j ∈ children of b on T_B do
      if j is chosen then
        continue; {The subtree rooted at j can only be chosen once for integration, as illustrated in Figure 3.}
      else
        identify vertex k in the subtree rooted at j such that arg max_k(M_AB(i, k)/β^distance(k,j));
        if M_AB(i, k) > max_score then
          max_score = M_AB(i, k);
          to_merge = k;
          to_merge_root = j;
        end if
      end if
    end for
    if to_merge = NULL then
      break;
    else
      save merge pair (i, to_merge) in T_AB;
      mark to_merge_root as chosen;
      push (i, to_merge) into queue q;
    end if
  end for
end while
return T_AB;
```

Algorithm 1: HeuristicMerge(T_A, T_B, M_{AB}).

conclude that such a brutal-force approach needs to pick up a best solution from an exponential number of solutions. The brutal-force approach is clearly not acceptable for integrating large ontologies and may not even work for ontologies with only a few dozens of vertices.

A heuristic solution can be developed by following an idea similar to the above brutal-force approach. However, instead of trying all possibilities, the heuristic solution will greedily merge vertices following the topological order. When selecting a matching vertex for vertex a from ontology T_A, the heuristic approach will greedily select a vertex b from allowable candidates in T_B and iteratively apply such selections to descendants of a. According to Criterion (1), if a is associated with b, then none of a's descendants are allowed to be associated with vertices other than b's descendants. In addition, if a's child a' is associated with b's child b' or its descendants, then none of a's other children are allowed to be associated with b' or its descendants any more. Given this, a greedy choice may very easily end up in a local optimum by choosing a best matching vertex at one step, while denying integrating opportunities of other vertices that may lead to a better final solution.

It is easy to see that the deeper a vertex being chosen for integration is, the more integration opportunities are lost. To alleviate such a situation, we propose a greedy approach by considering the relative depth (rdepth) of a chosen vertex with regard to an allowable vertex closest to the root. That is, given a vertex a from T_A, a vertex b from T_B is chosen when $M_{AB}(a, b)/\beta^{\text{rdepth}(b)}$ is maximized. When $\beta = 1$, the depth information does not take effective and when $\beta = \infty$, each vertex will only be associated with an allowable vertex closest to the root.

Algorithm 1 describes the pseudocode of the heuristic integration. It starts by integrating virtual roots of the two ontologies. After that, the integration will be carried out iteratively from top to bottom by following Criterion (1) and the heuristic strategy described above. In the empirical study, we will see that the heuristic algorithm works better when the depth information is considered. However, in terms of the overall cohesion score, it is no match for our optimal and approximation solutions as described below.

2.1.2. Optimal and Approximation Solutions. By dividing the integration of two trees into node merging and subtree integrations, we have identified optimal structures in the basic problem, as stated by Lemmas 1 and 2. These optimal structures make it possible for us to develop efficient algorithms (Algorithms 2 and 3) that achieve optimal or approximation solutions. In the following, we first describe the two important lemmas suggesting the optimal structures and their proofs before describing our proposed algorithms.

Lemma 1. *Let r_a be the root of tree T_A and r_b the root of tree T_B. Let T_{A-r_a} and T_{B-r_b} represent two sets of sub trees rooted at r_a of*

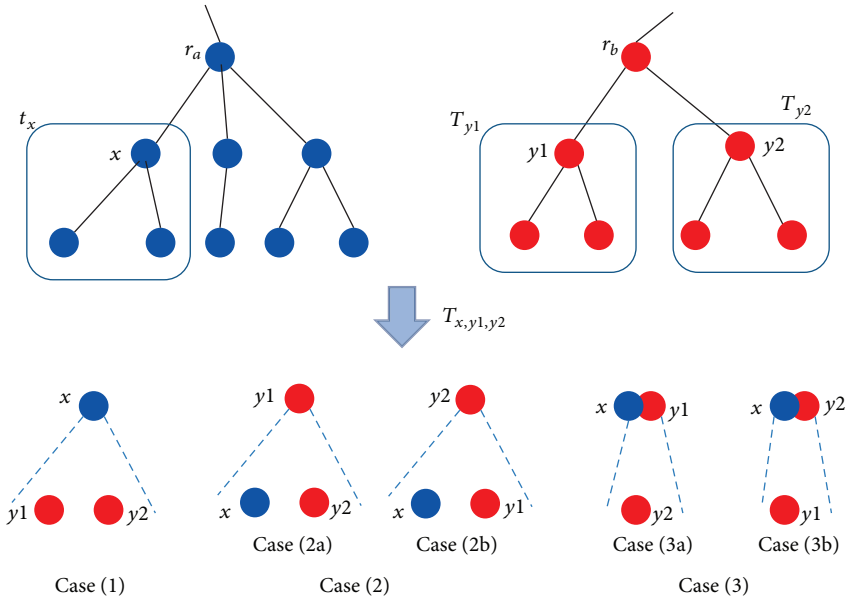


Figure 3: An illustration of three cases in the proof of Lemma 2.

```
Sort vertices in T_A and T_B in the topological order;
for i = |V(T_A)| to 0 do
  for j = |V(T_B)| to 0 do
    score = M_AB(i, j) + MaxMatch(T_{A−i}, T_{B−j});
    score_a = MaxMatch(T_{A−i}, T_j);
    score_b = MaxMatch(T_i, T_{B−j});
    cohesion_matrix(i, j) = max(score, score_a, score_b);
  end for
end for
return cohesion_matrix;
```

Algorithm 2: BuildCohesionMatrix(T_A, T_B).

T_A *and* r_b *of* T_B, *respectively.* T_{A-r_a} *does not include* r_a *and* T_{B-r_b} *does not include* r_b. *One has* $g(T_A, T_B) = \max(M_{AB}(r_a, r_b) + g(T_{A-r_a}, T_{B-r_b}), g(T_A, T_{B-r_b}), g(T_{A-r_a}, T_B))$.

Proof. We can divide the integration of tree T_A with tree T_B into two cases according to the merging of their roots.

(1) The roots of T_A and T_B are merged together.

(2) The roots of T_A and T_B are not merged together.

For case (1), it is clear that the cohesion score is $M_{AB}(r_a, r_b) + g(T_{A-r_a}, T_{B-r_b})$.

For case (2), we conclude that either T_A is integrated with T_{B-r_b} (r_b is out of integration) or T_B is integrated with T_{A-r_a} (r_a is out of integration). Otherwise, we will have a merged tree T_{AB} with two roots r_a and r_b, a contradiction to the fact that T_{AB} is a tree. Therefore, the cohesion score is either $g(T_A, T_{B-r_b})$ or $g(T_{A-r_a}, T_B)$.

Combining cases (1) and (2) and according to Criterion (2), we have

$$g(T_A, T_B) = \max\left(M_{AB}(r_a, r_b) + g\left(T_{A-r_a}, T_{B-r_b}\right), g\left(T_A, T_{B-r_b}\right), g\left(T_{A-r_a}, T_B\right)\right). \quad (1)$$

□

Lemma 2. *Let* T_{A-r_a} *and* T_{B-r_b} *represent two sets of trees obtained by removing the root vertices* r_a *from* T_{A-r_a} *and* r_b *from* T_{B-r_b}. *One has*

$$g\left(T_{A-r_a}, T_{B-r_b}\right) = \max\left(\sum_{(T_x, T_y)\in R}\left(g\left(T_x, T_y\right)\right)\right). \quad (2)$$

Here $T_x \in T_{A-r_a}$, $T_y \in T_{B-r_b}$, *and* R *is a matching of trees in* T_{A-r_a} *with trees in* T_{B-r_b}.

Proof. To prove this lemma, we first prove that for any tree $T_x \in T_{A-r_a}$, it can be integrated with no more than one tree in T_{B-r_b}. We will prove this claim by contradiction. Assume there are two trees $T_{y_1} \in T_{B-r_b}$ and $T_{y_2} \in T_{B-r_b}$ and they integrate with a tree $T_x \in T_{A-r_a}$ into an integrated tree T_{x,y_1,y_2}. There are three cases for the root r of T_{x,y_1,y_2}, as illustrated in Figure 3:

(1) r contains only the root of T_x;

(2) r contains only the root of T_{y_1} or T_{y_2};

(3) r contains the root of T_x and the root of T_{y_1} or T_{y_2}.

For case (1), the lowest common ancestor of the roots of T_{y_1} and T_{y_2} in the integrated tree T_{x,y_1,y_2} will no longer contain their lowest common ancestor in T_B, a contradiction to

```
push (0, 0, NULL) into queue q;
while |q| > 0 do
  (a, b, c) = pop(q);
  if a = −1 then
    make T_B(b) as a subtree of T_AB rooted at b; {T_B(b) is a sub tree of T_B rooted at b}
  else if b = −1 then
    make T_A(a) as a subtree of T_AB rooted at a; {T_A(a) is a sub tree of T_A rooted at a}
  else
    let d = (a, b) and make d a child of c on T_AB;
    score = M_AB(a, b) + MaxMatch(T_A−a, T_B−b);
    score_a = MaxMatch(T_A−a, T_b);
    score_b = MaxMatch(T_a, T_B−b);
    if score >= score_a && score >= score_b then
      push matching results of T_A−a, T_B−b with d into q;
    else if score_a >= score_b then
      push matching results of T_A−a, T_b with d into q;
    else
      push matching results of T_a, T_B−b with d into q;
    end if
  end if
end while
return T_AB;
```

Algorithm 3: BuildNewOnto(cohesion_matrix, T_A, T_B).

Criterion (1). For cases (2) and (3), the root of T_{y_2} (or the root of T_{y_1}) will be the descendant of the root of T_{y_1} (or the root of T_{y_2}) in the integrated tree T_{x,y_1,y_2}, a contradiction to Criterion (1). For integration involving more than two trees from T_{B-r_b}, we can still follow the above procedure to reach contradictions. Thus, the claim is proven.

Without loss of generality, we can see that, for any tree $T_y \in T_{B-r_b}$, it can be integrated with no more than one tree in T_{A-r_a}. Therefore, the integration between T_{A-r_a} and T_{B-r_b} corresponds to a matching in a weighted bipartite graph in which two sets of nodes represent trees from T_{A-r_a} and T_{B-r_b}, respectively, and edges represent corresponding cohesion scores. According to Criterion (2), $g(T_{A-r_a}, T_{B-r_b}) = \max(\sum_{(T_x,T_y)\in R}(g(T_x, T_y)))$ and we conclude that $g(T_{A-r_a}, T_{B-r_b})$ corresponds to the weight of a maximum weighted matching in the above bipartite graph. □

Given Lemma 2, we can see that the following corollary is correct.

Corollary 3. *Define* $MaxMatch(X, Y) = \sum_{(T_x,T_y)\in R} g(T_x, T_y)$, *where R is a maximum weighted matching of trees in forests X and Y given* $g(T_x, T_y)$ *for any tree pair* $(T_x, T_y) \in X \times Y$. *One concludes that, for any two forests* T_A *and* T_B, $g(T_A, T_B) = MaxMatch(T_A, T_B)$.

With Lemma 1 and Corollary 3, we are able to design an efficient dynamic programming algorithm achieving the global optimum for the ontology integration problem. The pseudocode for calculating the maximum cohesion score is described in Algorithm 2, which visits ontology vertices in reverse topological order when filling up the cohesion matrix.

At the end of Algorithm 2, the cohesion matrix is filled up with optimal cohesion scores and the maximum cohesion score is saved at entry (0, 0), as described by Theorem 4.

Theorem 4. *It holds that* $cohesion\_matrix(i, j) = g(T_{A-i}, T_{B-j})$ *and* $cohesion\_matrix(0, 0) = g(T_A, T_B)$.

Proof. We will prove this theorem by mathematical induction.

Let $|V(T_A)| = n$ and $|V(T_B)| = m$; it is easy to see that cohesion_matrix(n, m) is optimal because n and m correspond to leaf nodes in the topological order. Thus T_{A-n} and T_{B-m} are empty sets and cohesion_matrix$(n, m) = M_{AB}(n, m) = g(T_n, T_m)$.

When $i = n$ and $j < m$, according to Lemma 1, to integrate T_n with T_j, either T_n (the leaf vertex) is merged with T_j or T_n is integrated with T_{B-j}. The maximum score of integrating T_n with T_{B-j} is available at the time cohesion_matrix(n, j) is being calculated because of the reverse topological visit. Thus, according to both Lemma 1 and Corollary 3, we conclude that $g(T_n, T_j) = \max(M_{AB}(n, j), \text{MaxMatch}(T_n, T_{B-j})) =$ cohesion_matrix(n, j). Similarly, we can conclude that $g(T_i, T_m) = \max(M_{AB}(i, m), \text{MaxMatch}(T_{A-i}, T_m)) =$ cohesion_matrix(i, m).

When $i < n$ and $j < m$, again, according to Lemma 1 and Corollary 3, we have $g(T_i, T_j) = \max(\text{score}, \text{score}_a, \text{score}_b) =$ cohesion_matrix(i, j). Due to the reserve topological visit, MaxMatch(T_{A-i}, T_{B-j}), MaxMatch(T_{A-i}, T_j), and MaxMatch(T_i, T_{B-j}) are available at the time cohesion_matrix(i, j) is being calculated. □

Given the definition of $g(T_A, T_B)$, Theorem 4 in fact proves that the proposed approach achieves the global optimum. However, the global optimal solution is built upon

the maximum weighted matching (recall Corollary 3). As discussed in Section 2.1.3, the maximum weighted matching is time-consuming and therefore we propose an approximation solution in that section.

Although Algorithm 2 builds the cohesion matrix with optimal cohesion scores, it does not construct the integrated ontological knowledge structure. We may save the ontology integration details along with the cohesion scores. However, that will cost $O(n^3)$ memory space (assuming each ontology has $O(n)$ vertices) and significantly reduce the capacity of the algorithm in handling large ontology integrations. Quite interestingly, we find that it is not necessary to save the integration details in order to construct the integrated ontology. The construction can be done by a process reverse to the construction of cohesion matrix, as described in Algorithm 3.

Algorithm 3 uses the cohesion matrix constructed by Algorithm 2 and builds the integrated ontology tree still by following Lemma 1 and Corollary 3, but in a reverse way of Algorithm 2. The construction is performed in a Breadth-First fashion which uses a queue q to maintain triples. Each triple (a, b, c) is an association of three elements: a is the matched vertex from ontology T_A; b is the matched vertex from ontology T_B; and c is their parent on the merged ontology T_{AB}. By following the basic idea of the proof of Theorem 4 we can show that Algorithm 3 builds an optimal integrated ontology with the cohesion matrix provided from Algorithm 2. We omit the proof for succinctness.

2.1.3. Time Complexity Analysis and an Approximation Solution. Assume an ontology size is $O(n)$. The cohesion matrix has $O(n^2)$ entries to fill up. The computation for each entry is a matching whose time complexity depends on the implementation. The maximum weighted matching takes $O(n^3)$ using the famous Hungarian algorithm [21], and although it achieves optimum, it is too costly for large ontologies. The maximal weighted matching, however, takes $O(n^2 \log n)$ time and, more importantly, results in an overall $O(n^2 \log n)$ time complexity for Algorithm 2. The analysis is given in the following. For each matching, the algorithm will access previously filled entries and each entry will be accessed only once and be involved only once in a sorting of $O(\log n)$ time. This is because each entry corresponds to two vertices whose cohesion score will be accessed when calculating the cohesion score of their parents. Thus, we conclude that the total time complexity of calculating the cohesion matrix using maximal weighted matching is $O(n^2 + n^2 \log n) = O(n^2 \log n)$. Since building the new ontology has the same time complexity as building the cohesion matrix, this is also the total time complexity for integrating two ontologies by maximal weighted matching. The maximal weighted matching also has a guaranteed lower bound on the results. It achieves a (1/2)-approximation solution (i.e., the overall cohesion score will be at least 1/2 of the optimal cohesion score) as pointed out in [22].

Since the maximal weighted matching results in an overall good performance on time complexity and approximation rate, we used the maximal weighted matching in our empirical study for Algorithms 2 and 3. Readers may also choose other matching algorithms (such as the one described in [22]) to achieve slightly better approximation rates. However, the weighted matching is a replaceable module for our algorithms and it is not the focus of this work to build a fast and close-to-optimal weighted matching algorithm.

Compared to the time complexity of integrating ontologies by the dynamic programming approach as described in Algorithms 2 and 3, the heuristic approach described at the beginning of this section also has $O(n^2)$ in the worst case. However, we conjecture that the heuristic approach has a much smaller average time complexity because, in each step, the heuristic approach may exclude a large number of matching opportunities.

2.2. Integrating Multiple Ontologies. In the previous section we proposed methods for integrating two ontologies. In some biomedical applications [23, 24], we are interested in the associations involving more than two objects. Integration of multiple ontologies of these objects will generate an innovative view on these complex relationships. Similar to the basic problem formulation, we can formulate the multiple ontology integration as follows.

Given k ontology trees $T_1, T_2, \ldots, T_k$ and a closeness matrix M_{ij} for any two trees T_i and T_j, how can we efficiently generate an integrated ontology tree $T_{1,2,\ldots,k}$ meeting the following criteria?

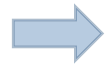
(1) For any two vertices x and y in a tree T_i, their lowest common ancestor $\mathrm{LCA}_{T_i}(x, y)$ is contained by $\mathrm{LCA}_{T_{1,2,\ldots,k}}(x, y)$.

(2) It holds that $\mathrm{argmax}_{T_{1,2,\ldots,k}} f(T_{1,2,\ldots,k}) = \sum_{X \in V(T_{AB})} \sum_{u \in X, v \in X, \sigma(u) < \sigma(v)} M_{\sigma(u),\sigma(v)}(u, v)$. Here $M_{\sigma(u),\sigma(v)}(u, v)$ is the entry value in the closeness matrix for two vertices u and v (one from tree $T_{\sigma(u)}$ and the other from tree $T_{\sigma(v)}$) contained in the node X. For a vertex v from an original ontology, $\sigma(v)$ is defined as its original ontology ID.

Again, we name the function $f_{T_{1,2,\ldots,k}}$ the cohesion of the integrated ontology $T_{1,2,\ldots,k}$. For each node X in the integrated ontology, we define its weight as $weight(X) = \sum_{u \in X, v \in X, \sigma(u) < \sigma(v)} M_{\sigma(u),\sigma(v)}(u, v)$. Correspondingly, we define function $g(T_1, T_2, \ldots, T_k) = \max_{T_{1,2,\ldots,k}} (\sum_{X \in V(T_{1,2,\ldots,k})} weight(X))$ as the maximum cohesion function for integrating the ontologies $T_1, T_2, \ldots, T_k$. As we can see in the above formulation, the overall cohesion score of integration is the summed weight of each node, which is the sum of pairwise closeness scores.

The formulation of multiple ontology integration is similar to the basic version, and it is not difficult to show that optimal structures described in Lemmas 1 and 2 can be extended to a high dimension. However, the extension of algorithms described in Section 2.1.2 for integrating two ontologies is not feasible for solving the multiple ontology integration. This is because if we need to extend Algorithms 2 and 3 to this problem, we need to build a cohesion matrix of k dimensions. It implies that we need at least $O(n^k)$ operations to fill up the score matrix assuming the size of an ontology

	A	B	C	D
A	—	10	6	8
B	10	—	7	5
C	6	7	—	9
D	8	5	9	—



	AB	C	D
AB	—	*	*
C	*	—	9
D	*	9	—

(*) Entries to update

Figure 4: An example of the InterOntology matrix's change at the first iteration in Algorithm 4 for integrating four ontologies.

build the $k \times k$ InterOntology matrix;
for $i = 1$ to $k - 1$ **do**
  identify the active tree pair $\langle T_X, T_Y \rangle$ that corresponds to the highest score in the InterOntology matrix;
  integrate T_X and T_Y into $T_{X,Y}$;
  mark T_X and T_Y as inactive;
  update relationship matrices;
  update InterOntology matrix;
end for
return $T_{1,2,\ldots,n}$;

Algorithm 4: GreedyMultiInt($\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$).

is $O(n)$. This is clearly not acceptable for high dimensional ontology integration.

2.2.1. Greedy Approach. From the above discussion we can see that direct extension of Algorithms 2 and 3 for integrating two ontologies is practically not feasible for integrating a large number of ontologies. However, we can still use these algorithms for integrating multiple ontologies, by iteratively integrating two ontologies and generating a new closeness matrix. Given the ontologies $T_1, T_2, \ldots, T_k$, we can first integrate T_1 and T_2 into $T_{1,2}$ and then build the closeness matrix between $T_{1,2}$ and T_3 using the relationship matrices between T_1 and T_2 and between T_1 and T_3. Specifically, assume X is a node on the integrated ontology $T_{1,2}$, and X contains a vertex a from T_1 and a vertex b from T_2. Then, the entry (X, c) of the closeness matrix between $T_{1,2}$ and T_3 is $M_{T_{1,2},T_3}(X, c) = M_{T_1,T_3}(a, c) + M_{T_2,T_3}(b, c)$. After the new closeness matrix is generated, we can continue integrating $T_{1,2}$ and T_3 into $T_{1,2,3}$ and generating another new closeness matrix. We will eventually get the integrated ontology $T_{1,2,\ldots,k}$ by repeating the above process. To facilitate the following discussions, we name the above approach the *basic multiple integration approach*.

Although the basic multiple integration approach can finish integrating multiple ontologies, it blindly integrates ontologies without using any cohesion information between ontologies that may lead to a better integration result. To improve the basic multiple integration, we propose a greedy approach that uses the cohesion information between ontologies to guide the integration. The basic steps of the greedy approach are outlined in Algorithm 4. To facilitate the understanding of Algorithm 4, we use Figure 4 to illustrate an example of the InterOntology matrix's change at the first iteration of integrating four ontologies A, B, C, and D.

The key idea in Algorithm 4 is to maintain an InterOntology matrix which guides the integration. Initially, this matrix is filled with the overall cohesion score of every pair of ontologies. In each step, this matrix is updated with overall cohesion scores between newly integrated ontology and existing active ontologies. The integration will take place between two active ontologies which have the highest score in the InterOntology matrix.

When we used the overall cohesion score between two ontologies to update the InterOntology matrix, we observed an interesting phenomenon that the integration in most cases is a process continuously expanding an integrated ontology. Consequently, the greedy approach is likely to yield a result similar to the basic approach.

This phenomenon can be explained by the definition of maximum cohesion function, which takes into account all pairwise closeness between merged terms. Thus, the more ontologies contained in an integrated ontology are, the more likely it will have high overall cohesion scores with other ontologies. As a result, it creates unfairness for the integration selection. To fix this issue, we use the adjusted overall cohesion scores in updating the InterOntology matrix as follows.

Given an ontology T_X and an ontology T_Y where X and Y are nonempty sets of ontology IDs, we define the adjusted cohesion score between T_X and T_Y as $AdjCoh(T_X, T_Y) = \sum_{Z \in V(T_{XY})} \sum_{x \in Z, y \in Z, \sigma(x) \in X, \sigma(y) \in Y} M_{\sigma(x),\sigma(y)}(x, y)/|X||Y|$, where T_{XY} is the integrated ontology built by Algorithms 2 and 3. The adjusted cohesion score is in fact the weight increase by integrating T_X and T_Y, divided by the size of X times the size of Y; that is, $AdjCoh(T_X, T_Y) = \sum_{Z \in V(T_{XY})}(weight(Z)) - \sum_{X \in V(T_X)}(weight(X)) - \sum_{Y \in V(T_Y)} weight(Y)/|X||Y|$. For each node merging, closeness scores will be added to the total weight when the merging takes place between vertices from ontology set X and vertices from ontology set Y. Thus, the weight increase by integrating T_X and T_Y is proportional to the number of ontologies in X times the number of ontologies in Y, and consequently we averaged the weight increase by $|X||Y|$.

2.2.2. Fast Approximation Algorithm. Although the basic multiple integration and the greedy multiple integration approaches discussed above are able to integrate multiple ontologies, none of them provide any guarantee on the results

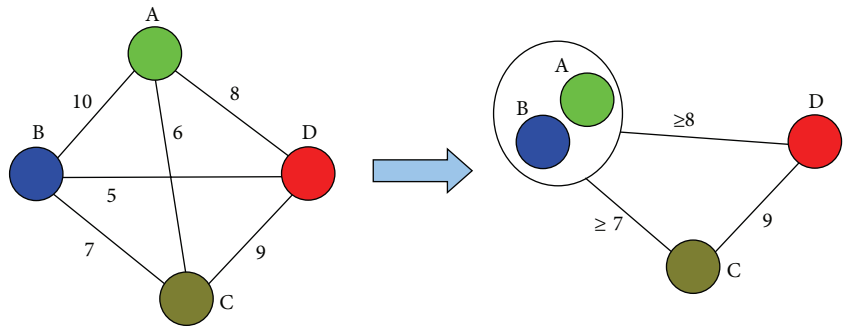


Figure 5: An illustration of vertex/edge contraction and weight updates in an iteration of Algorithm 5. Each vertex represents an ontology.

```
for i = 1 to k − 1 do
  for j = i + 1 to k do
    push ⟨g(T_i, T_j), i, j⟩ into a set S ordered by the first element in descending order;
  end for
end for
while |S| > 0 do
  ⟨z, x, y⟩ = pop(S);
  if adding (x, y) does not form a cycle in 𝒢 then
    adding (x, y) to 𝒢;
    integrate T_C(x) and T_C(y) into T_C(x)∪C(y); {C(v) is a set of vertices including v that form a connected component in 𝒢.}
  end if
end while
return T_1,2,...,n;
```

Algorithm 5: FastMultiInt($\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$).

in comparison with the optimal solutions. By studying the maximum cohesion scores between ontologies under a graph setting, we identified an approximation structure and developed an approximation algorithm for integrating multiple ontologies. We name it fast approximation algorithm because it not only has a lower bound on the results, but also runs faster than the greedy multiple integration algorithm proposed above.

The fast approximation algorithm for integrating multiple ontologies is sketched in Algorithm 5. It only calculates the maximum cohesion score between every pair of ontologies once during the initial stage, and uses this information throughout the integration process even after it becomes stale. More importantly, this approach not only saves the time for recalculating the maximum cohesion scores, but also provides a lower bound guarantee as stated in Theorem 5, whose correctness is built on two important lemmas (Lemmas 6 and 7) which will be described subsequently.

Theorem 5. *The tree weight of the integrated tree $T_{1,2,\ldots,k}$ obtained by FastMultiInt algorithm is at least $1/(k-1)$ the weight of the optimal solution.*

Proof. We will use Lemmas 6 and 7 to prove this theorem. The proofs of Lemmas 6 and 7 are provided after the proof of this theorem. To facilitate the proof of this theorem, we build a fully connected weighted graph $\mathcal{G}$ in which each node corresponds to a tree for integration, and the weight of each edge corresponds to the weight increase (initially, this is the cohesion score) for integrating the corresponding trees. According to Lemma 6, $g(T_1, T_2, \ldots, T_k)$ (i.e., optimal cohesion score) is no more than the summed weight of edges in $\mathcal{G}$ (Claim 1).

Given $\mathcal{G}$, the integration by Algorithm 5 is a process of $k-1$ node contractions. After each contraction, the adjacent edge weights (cohesion scores) will be updated accordingly. According to Lemma 7, the weight of an updated edge will only increase over (or at least remain the same as) the maximum weight of the two contracted edges (Figure 5 provides an illustration of vertex/edge contraction and weight updates.) Thus, the overall cohesion score of the integration by Algorithm 5 is no less than the weight of the maximum spanning tree of $\mathcal{G}$ (Claim 2).

It is easy to see that the weight of a maximum spanning tree is no less than $1/(k-1)$ of the summed edge weight of $\mathcal{G}$, given the simple observation that each edge in $\mathcal{G}$ is either an edge of the maximum spanning tree or adjacent to an edge of the maximum spanning tree with an equal or heavier weight (Claim 3).

Combining Claims 1, 2, and 3, we complete the proof of this theorem. □

Lemma 6. *It holds that $g(T_1, T_2, \ldots, T_k) \leq \sum_{1 \leq i < j \leq k} g(T_i, T_j)$.*

Proof. According to the problem definition,

$$\begin{aligned} & g(T_1, T_2, \ldots, T_k) \\ & = \max\left(\sum_{X \in V(T_{1,2,\ldots,k})} weight(X)\right) \\ & = \max\left(\sum_{X \in V(T_{1,2,\ldots,k})} \sum_{u \in X, v \in X, \sigma(u) < \sigma(v)} M_{\sigma(u),\sigma(v)}(u, v)\right) \\ & = \left(\sum_{1 \le i < j \le k} f_{T_{1,2,\ldots,k}}(T_{i,j})\right) \le \sum_{1 \le i < j \le k} g(T_i, T_j). \end{aligned} \quad (3)$$

$f_{T_{1,2,\ldots,k}}(T_{i,j})$ is the cohesion score of $T_{i,j}$ whose integration is induced from $T_{1,2,\ldots,k}$. □

Lemma 7. *It holds that* $g(T_{P,Q}, T_S) - f(T_{P,Q}) \ge \max(g(T_P, T_S), g(T_Q, T_S))$.

Proof. According to the problem definition, integrating $T_{P,Q}$ with T_S will result in an integrated tree T_U where $U = P \cup Q \cup S$, and $g(T_{P,Q}, T_S) = \max_{T_U}(\sum_{X \in V(T_U)} weight(X)) = \max_{T_U}(f(T_{P,S}) + f(T_{Q,S}) + f(T_{P,Q}))$, where $T_{P,S}$, $T_{Q,S}$, and $T_{P,Q}$ are induced from T_U. Since $T_{P,Q}$ has been determined, we have $g(T_{P,Q}, T_S) = \max_{T_U}(f(T_{P,S}) + f(T_{Q,S})) + f(T_{P,Q})$. Without loss of generally, let us assume $\max(f(T_{P,S})) \ge \max(f(T_{Q,U}))$. Then, by restricting the integration between T_P and T_S in T_U following the integration that leads to $\text{argmax}_{T_{P,S}} f(T_{P,S})$, we will get a cohesion score no less than $g(T_P, T_S)$. Thus, we complete the proof for $g(T_{P,Q}, T_S) - f(T_{P,Q}) \ge \max(g(T_P, T_S), g(T_Q, T_S))$. □

2.2.3. Time Complexity Analysis. For the fast approximation algorithm (Algorithm 5), the time complexity for generating graph $\mathcal{G}$ (calculating the overall cohesion score for every pair of ontologies) is $O(k^2 n^2 \log n)$, assuming we use maximal weighted matching. Each integration will take $O(n^2 \log n)$ with an update of at most k closeness matrices which takes $O(k * n^2)$. There are at most k integrations; therefore the total time complexity is still $O(k^2 n^2 \log n)$.

Following the above analysis, we conclude that the time complexity of the greedy multiple integration algorithm is the same as the fast approximation algorithm. However, it requires updating of InterOntology matrix which takes an excessive $O(k^2 n^2 \log n)$ time. The empirical study shows that the fast approximation algorithm is much faster than the greedy multiple integration algorithm.

Finally, it is easy to see that the basic multiple integration approach takes $O(kn^2 \log n)$ time and is the fastest, but its overall cohesion scores are the worst as we will see in the empirical study.

2.2.4. Limitations. Integrating multiple ontologies may face two potential problems in real applications. First, how can we efficiently generate a closeness matrix for every pair of ontologies to be integrated? Our current method kDLS or onGrid is efficient for generating the closeness matrix for one pair of ontologies in most cases, but not efficient enough for generating closeness matrices for many pairs of ontologies. Second, not every pair of ontologies can be meaningfully integrated. It remains a problem to efficiently identify the feasibility of integrating a pair of ontologies. Therefore, the main purpose of Section 2.2 is to demonstrate that our proposed approach can be extended to integrate multiple ontologies, and we use synesthetic datasets in Section 3.3 to study the performance of algorithms proposed in Section 2.2.

3. Results and Discussion

We would like to study the performances of the proposed ontology integration methods by experiments on both real and synthetic datasets. We implemented five approaches in C++:

(1) Heuristic: heuristic approach for integrating two ontologies as described in Section 2.1.1;

(2) Approximate: approximate approach (Algorithms 2 and 3) with maximal weighted matching for guaranteeing the (1/2)-approximation rate;

(3) Basic: basic multiple integration approach as described in Section 2.2.1;

(4) Greedy: greedy multiple integration approach (Algorithm 4);

(5) FastApproximate: fast approximation multiple integration approach (Algorithm 5).

In the following, we report our study on the performances of (1) and (2) for integrating two ontologies on real datasets and (3), (4), and (5) for integrating multiple ontologies on synthetic datasets. All the experiments are carried out on a Linux cluster with 2.4 GHz AMD Opteron processors.

3.1. Integrating a Pair of Ontologies. The knowledge of drug-gene relationships is desirable in many pharmacology applications [25, 26]. By integrating the gene ontology and the drug ontology, we will be able to obtain rich information on the associations between drugs and genes under the ontology structures. Thus, in this set of experiments, we simulate real world knowledge discovery applications by integrating two real ontologies, gene ontology (GO) and National Drug File Reference Terminology (NDFRT). Both were obtained from the Unified Medical Language System (version: 2012AA). The closeness matrices between GO terms and drug terms were generated using onGrid [27] with a 4-neighborhood broadcast range (i.e., $k = 4$ with regard to [7]). onGrid follows the kDLS approach [7] and measures the closeness between two concepts based on the discovered paths (with length greater than one) between them. However, unlike kDLS, onGrid takes into consideration of concept semantic types in the closeness measurement. In the study performed in [27], the advantages of onGrid over kDLS are well illustrated.

The overall cohesion scores of Heuristic and Approximate on integrating GO and NDFRT are listed

Table 1: Cohesion scores of integrating real datasets.

Depth	GO term number	NDFRT term number	Cohesion scores			
			Heuristic $\beta = 1$	Heuristic $\beta = 6$	Heuristic $\beta = 100$	Approximate
3	66	6004	0.0505331	0.229958	0.21999	1.24696
4	710	6972	0.290392	0.284363	1.46585	9.3835
5	5355	14582	0.285923	1.37714	0.528289	33.9056
6	16231	32841	0.307941	0.341588	0.673406	74.0293

Table 2: Top 5 matched terms by the Approximate algorithm (depth = 6).

Rank	GO terms	NDFRT terms	Closeness score
1	C1135918 smooth muscle contractile fiber	C0282606 muscle neoplasms	1.63205
2	C0010813 cytokinesis	C0086376 GTP-binding proteins	1.15967
3	C0027747 axon terminus	C0030584 parovarian cyst	1.13352
4	C1155065 T cell activation	C0007082 carcinoembryonic antigen	1.00879
5	C0007595 cell growth	C0294028 BRCA2 protein	0.945284

Table 3: Top 5 matched terms by the Heuristic algorithm (depth = 6, $\beta = 100$).

Rank	GO terms	NDFRT terms	Closeness score
1	C0031845 biological_process	C0042890 VITAMINS	0.244862
2	C1166607 cellular_component	C1657248 apoptosome	0.142857
3	C0027540 tissue death	C0065932 MENADIOL	0.0434219
4	C0025519 metabolic process	C0042849 VITAMIN B	0.0370248
5	C0030012 oxidation-reduction process	C0027996 NICOTINIC ACID	0.0327109

in Table 1. To observe the integration over the ontology size change, in each experiment we use the ontology tree structure from the root to the specified depth (first column in Table 1) for integration. The sizes of the ontology terms involved in the integration are listed in the second and third columns of Table 1.

Recall in Section 2.1.1; $\beta^{\text{rdepth}(b)}$ is used to regulate the selection of vertices from high depths. Thus, we tested the Heuristic under $\beta = 1$ (depth information is nullified) and $\beta = 100$ (the vertex depth plays a critical role in the selection). Since nonleaf vertices in these datasets have around 6 children on average, we heuristically add a set of experiments by setting $\beta = 6$ so that $\beta^{\text{rdepth}(b)}$ will be close to the number of vertices excluded from the future integration.

From Table 1, we can see that Heuristic performs better when using the depth information to regulate the selection of vertices. However, Approximate is much better than Heuristic at all settings. Compared to the best cohesion scores of Heuristic in each row of Table 1, Approximate constructs an integrated ontology with the overall cohesion score ranging from 5.4 times to 109.9 times that of Heuristic. This clearly demonstrates the effectiveness of the proposed Approximate approach. Nevertheless, the heuristic approach has a much faster average running time as a result of excluding a large number of matching opportunities in each step.

Although the running time of Approximate is longer than Heuristic, it takes less than two hours to finish integrating two ontologies with about $16k$ and $33k$ vertices. Most of the biomedical ontologies are smaller than or similar to these sizes and Approximate approach will benefit the association study of these ontologies. For extremely large ontology pairs in which Approximate is unable to finish the integration within a reasonable time, Heuristic may provide a quick view on their integration.

3.2. Understanding the Merged Ontology Terms. To understand what terms are merged in integrating real ontologies, we use the integration of GO and NDFRT at depth 6 as an example. Tables 2 and 3 list the top 5 pairs of merged terms (sorted by their closeness scores) by Heuristic and Approximate, respectively. As mentioned above, these scores are from the closeness matrix generated by onGrid based on the discovered paths between them. For example, "C1155065:T-Cell Activation – – is_physiologic_effect_of_chemical_or_drug – – > C0393002: Carcinoembryonic Antigen Peptide 1 – – has_target – – > C0007082:Carcinoembryonic Antigen" is such a path.

From Tables 2 and 3 we can observe that the Approximate algorithm merges terms with much higher similar scores than the Heuristic algorithm. Quite interestingly, we observed that the top ranked merging in Table 3 is between "biological_process" and "VITAMINS." The "biological_process" is an abstract term which is very close to the root of the GO ontology. Such a fact suggests that the top level terms will likely preempt the merging choices over their descendants. As a result of this greedy approach,

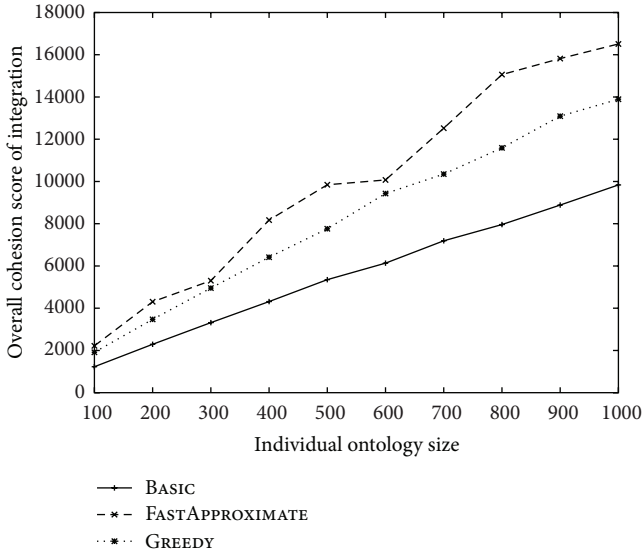


Figure 6: The change of overall cohesion score over the increase of the size of each ontology. The number of ontologies is fixed at 10.

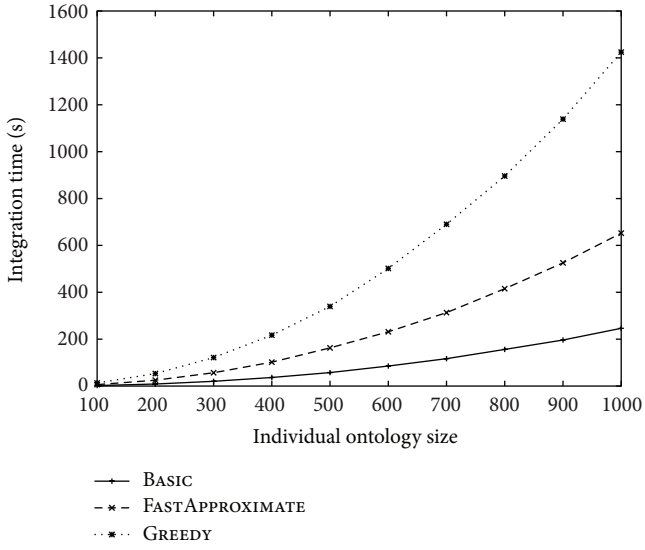


Figure 8: The change of integration time over the increase of the size of each ontology. The number of ontologies is fixed at 10.

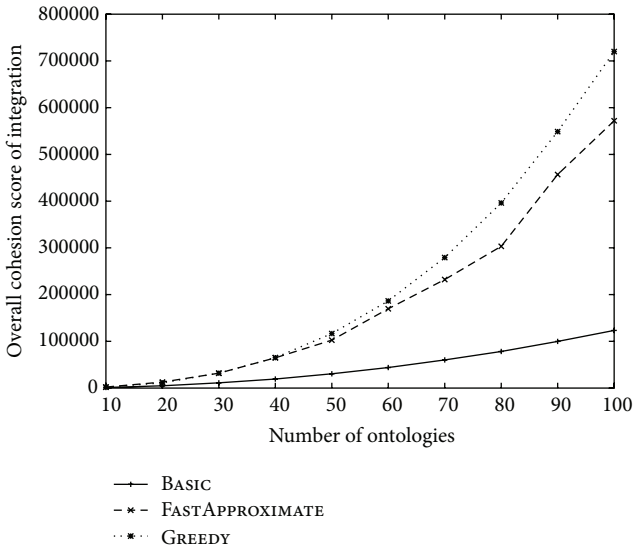


Figure 7: The change of overall cohesion score over the increase of the number of ontologies. The size of each ontology is fixed at 100.

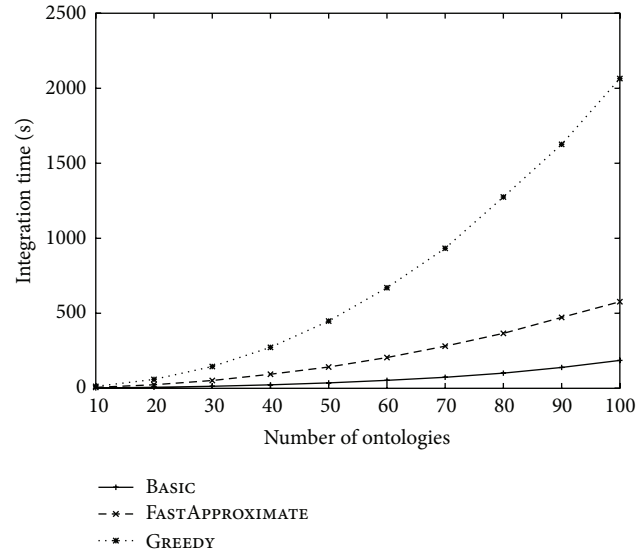


Figure 9: The change of integration time over the increase of the number of ontologies. The size of each ontology is fixed at 100.

the Heuristic algorithm will end at a local optimum which is far from being optimal.

A snapshot of ontology integration by Approximate as shown in Figure 10 provides a good insight on the algorithm work. In each bracket of two merged terms, the left part is the closeness score and the right part is the cohesion score. We can observe that most closeness scores are zero or close to zero, while the corresponding cohesion scores are much higher. This is understandable because the snapshot is primarily on the top level terms of both ontologies. For these terms, they have a large number of subclass (descendant) terms, and optimizing the integration of their subclass terms far outweighs integrating of themselves. The result of such integration provides novel knowledge of association between ontology terms. That is, even if two terms are not that close according to some closeness measurement, they can be structurally associated under their ontology context. For example, the GO term "biological_process" is merged with the NDFRT term "chemical ingredients"; even their closeness score is zero from the onGrid output. However, such integration is interesting because it shows that the merging is trying to link the chemical compounds with the biological/cellular processes so that corresponding associations between the cellular processes and chemical structures can be established. This demonstrates the purpose of integrating two ontologies, that is, identifying associations with respect to both term similarities and structural contexts.

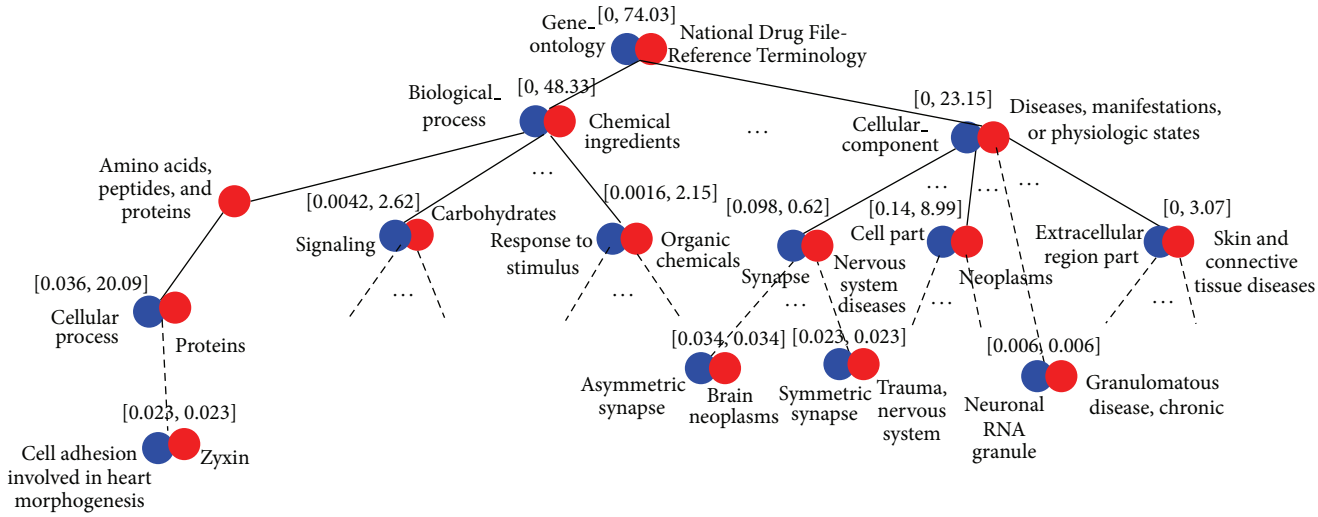


Figure 10: A snapshot of ontology integration between GO and NDFRT by Approximate (depth = 6). Blue nodes are terms from GO and red nodes are terms from NDFRT. Solid lines are edges and dashed lines are paths consisting of 1 or more edges. In each bracket, the left part is the closeness score and the right part is the cohesion score.

In fact, there are multiple studies to justify the structural associations seen in Figure 10, such as the association between "signaling" and "carbohydrates" [28] and the association between "extracellular region part" and "skin and connective tissue diseases" [29].

In addition, we have noticed a number of meaningful integrations between GO terms and neurological terms in the NDFRT. For example, synapse is a brain related structure and the term "symmetric synapse" is associated with "trauma," and the term "asymmetric synapse" is associated with "brain neoplasms." Similarly, it is reasonable to see that "neuronal RNA granule" is integrated with "granulomatous disease," a granule associated disease. As another example, it is very interesting to notice that "zyxin" is associated with "cell adhesion involved in heart morphogenesis" and that provides a link with the formation of heart.

The above observations suggest a novel way of using our ontology integration method to perform association studies between biomedical concepts.

3.3. Integrating Multiple Ontologies. In the following experiments we will study the performances of Basic, Greedy, and FastApproximate in integrating multiple ontologies. All the three approaches are built upon Approximate, which performs very well in the previous study for integrating two real ontology datasets.

We use two sets of synthetic datasets in this study. In the first set of datasets, we fix the number of ontologies to be 10 and vary the size of each ontology from 100 to 1000. In the second set of datasets, we fix the size of each ontology to be 100 and vary the number of ontologies from 10 to 100. All the ontologies are randomly generated by constructing a minimal spanning tree from a random matrix. The relationship matrix between every pair of ontologies is also randomly generated with entry values ranging from 0 to 1. For each experiment, we generate 10 random datasets and the results reported in the following are the average results over the 10 random datasets.

The overall cohesion scores of the three approaches over different ontology sizes and over different numbers of ontologies were reported in Figures 6 and 7, respectively. FastApproximate outperforms all the other approaches in Figure 6, which is consistent with the analysis of its approximation rate. However, Greedy slightly outperforms FastApproximate in Figure 7 especially when the ontology number is large. This is understandable because when the number of ontology (k) increases, the approximation rate (as stated in Theorem 5) decreases and becomes less significant. This result also justifies the choice of adjusted cohesion score for Greedy as described at the end of Section 2.2.1.

The integration time of the three approaches over different ontology sizes and over different numbers of ontologies was reported in Figures 8 and 9, respectively. These figures are consistent with the time complexity analysis given in Section 2.2.3. In particular, we noticed that the integration time of Greedy deteriorates sharply over the ontology number increase. In contrast, FastApproximate is much more scalable and has a time curve similar to Basic.

These results suggest that FastApproximate has the best overall performance in integrating multiple ontologies.

4. Conclusions

In this work, we started with a basic problem on integrating a pair of ontology tree structures with a given closeness matrix, and later we advanced the basic problem to the problem of integrating large number of ontologies. We proved optimal structures in the basic problem and developed both optimal and efficient approximation solutions. Although the multiple ontology integration problem has similar optimal structures, it is not feasible to extend the optimal and efficient approximation solutions for the basic problem to efficiently handle

multiple ontology integration. To tackle the challenge of integrating a large number of ontologies, we developed both an effective greedy approach and a fast approximation approach. The empirical study not only confirms our analysis on the efficiency of the proposed method, but also demonstrates that our method can be used effectively for biomedical association studies.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] D. L. McGuinness, F. van Harmelen, and M. K. Smith, Owl web ontology language overview. W3C recommendation, 10(2004-03):10, 2004.

[2] O. Bodenreider, "The unified medical language system (UMLS): integrating biomedical terminology," *Nucleic Acids Research*, vol. 32, supplement 1, pp. D267–D270, 2004.

[3] X. Wu, R. Jiang, M. Q. Zhang, and S. Li, "Network-based global inference of human disease genes," *Molecular Systems Biology*, vol. 4, no. 1, 2008.

[4] B. Linghu, E. S. Snitkin, Z. Hu, Y. Xia, and C. DeLisi, "Genome-wide prioritization of disease genes and identification of disease-disease associations from an integrated human functional linkage network," *Genome Biology*, vol. 10, no. 9, article R91, 2009.

[5] D. Botstein and N. Risch, "Discovering genotypes underlying human phenotypes: past successes for mendelian disease, future approaches for complex disease," *Nature Genetics*, vol. 33, pp. 228–237, 2003.

[6] H. S. Pinto and J. P. Martins, "A methodology for ontology integration," in *Proceedings of the 1st International Conference on Knowledge Capture (K-CAP '01)*, pp. 131–138, ACM, 2001.

[7] Y. Xiang, K. Lu, S. L. James, T. B. Borlawsky, K. Huang, and P. R. O. Payne, "K-Neighborhood decentralization: a comprehensive solution to index the UMLS for large scale knowledge discovery," *Journal of Biomedical Informatics*, vol. 45, no. 2, pp. 323–336, 2012.

[8] J. Dutkowski, M. Kramer, M. A. Surma et al., "A gene ontology inferred from molecular networks," *Nature Biotechnology*, vol. 31, no. 1, pp. 38–45, 2013.

[9] R. Navigli, P. Velardi, and A. Gangemi, "Ontology learning and its application to automated terminology translation," *IEEE Intelligent Systems*, vol. 18, no. 1, pp. 22–31, 2003.

[10] J. Jannink and G. Wiederhold, "Thesaurus entry extraction from an on-line dictionary," in *Proceedings of the Fusion Conference*, Sunnyvale, Calif, USA, 1999.

[11] C. Papatheodorou, A. Vassiliou, and B. Simon, "Discovery of ontologies for learning resources using word-based clustering," in *World Conference on Educational Multimedia, Hypermedia and Telecommunications*, pp. 1523–1528, 2002.

[12] D. L. Rubin, M. Hewett, D. E. Oliver, T. E. Klein, and R. B. Altman, "Automating data acquisition into ontologies from pharmacogenetics relational data sources using declarative object definitions and xml," *Pacific Symposium on Biocomputing*, pp. 88–99, 2001.

[13] N. F. Noy, "Semantic integration: a survey of ontology-based approaches," *ACM SIGMOD Record*, vol. 33, no. 4, pp. 65–70, 2004.

[14] S. Abels, L. Haak, and A. Hahn, "Identification of common methods used for ontology integration tasks," in *Proceedings of the 1st International Workshop on Interoperability of Heterogeneous Information Systems (IHIS '05)*, pp. 75–78, ACM, November 2005.

[15] A. Gangemi, D. Pisanelli, and G. Steve, "Ontology integration: experiences with medical terminologies," in *Formal Ontology in Information Systems*, vol. 46, pp. 98–94, IOS Press, Amsterdam, The Netherlands, 1998.

[16] N. F. Noy and M. A. Musen, "SMART: automated support for ontology merging and alignment," in *Proceedings of the 12th Workshop on Knowledge Acquisition, Modelling, and Management (KAW '99)*, Banff, Canada, 1999.

[17] A. Doan, J. Madhavan, P. Domingos, and A. Halevy, "Learning to map between ontologies on the semantic web," in *Proceedings of the 11th International Conference on World Wide Web (WWW '02)*, pp. 662–673, ACM, May 2002.

[18] J. Xie, F. Liu, and S.-U. Guan, "Tree-structure based ontology integration," *Journal of Information Science*, vol. 37, no. 6, pp. 594–613, 2011.

[19] D. Calvanese, G. De Giacomo, and M. Lenzerini, "A framework for ontology integration," in *The Emerging Semantic WebSelected Papers from the First Semantic Web Working Symposium*, pp. 201–214, 2002.

[20] O. Udrea, L. Getoor, and R. J. Miller, "Leveraging data and structure in ontology integration," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '07)*, pp. 449–460, ACM, June 2007.

[21] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 1-2, pp. 83–97, 1955.

[22] D. E. Drake Vinkemeier and S. Hougardy, "A linear-time approximation algorithm for weighted matchings in graphs," *ACM Transactions on Algorithms*, vol. 1, no. 1, pp. 107–122, 2005.

[23] A. J. J. Wood, W. E. Evans, and H. L. McLeod, "Pharmacogenomics—drug disposition, drug targets, and side effects," *The New England Journal of Medicine*, vol. 348, no. 6, pp. 538–549, 2003.

[24] R. B. Altman, "Pharmgkb: a logical home for knowledge relating genotype to drug response phenotype," *Nature Genetics*, vol. 39, no. 4, article 426, 2007.

[25] R. B. Kim, D. O'Shea, and G. R. Wilkinson, "Interindividual variability of chlorzoxazone 6-hydroxylation in men and women and its relationship to CYP2E1 genetic polymorphisms," *Clinical Pharmacology & Therapeutics*, vol. 57, no. 6, pp. 645–655, 1995.

[26] T. N. Ferraro and R. J. Buono, "The relationship between the pharmacology of antiepileptic drugs and human gene variation: an overview," *Epilepsy and Behavior*, vol. 7, no. 1, pp. 18–36, 2005.

[27] A. Albin, X. Ji, T. B. Borlawsky et al., "Enabling online studies of conceptual relationships between medical terms: developing an efficient web platform," *JMIR Medical Informatics*, vol. 2, no. 2, article e23, 2014.

[28] S. Chandrasekaran, J. W. Dean III, M. S. Giniger, and M. L. Tanzer, "Laminin carbohydrates are implicated in cell signaling," *Journal of Cellular Biochemistry*, vol. 46, no. 2, pp. 115–124, 1991.

[29] J. Uitto and D. Kouba, "Cytokine modulation of extracellular matrix gene expression: relevance to fibrotic skin diseases," *Journal of Dermatological Science*, vol. 24, pp. S60–S69, 2000.